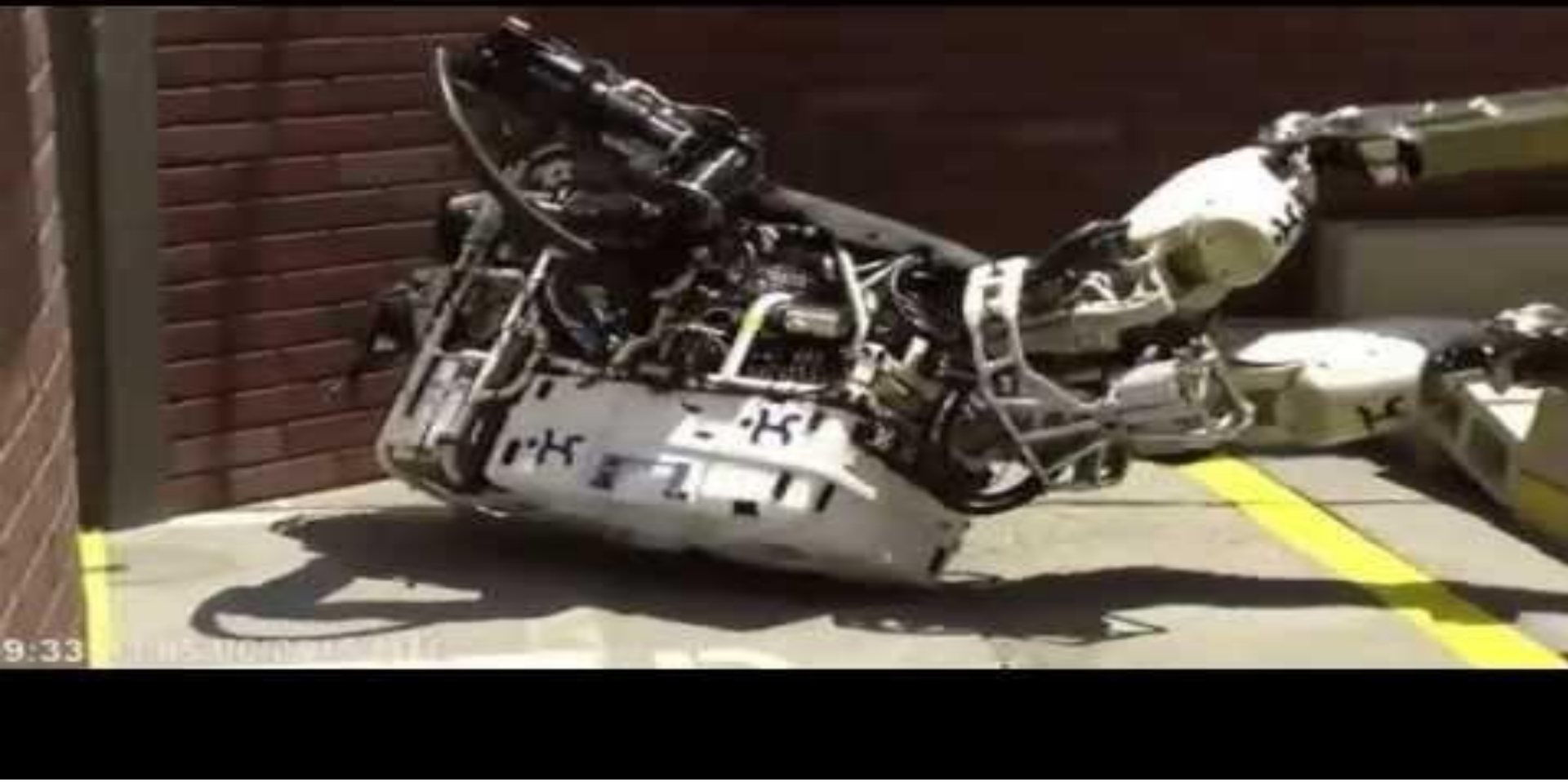
Sissejuhatus
Infotehnoloogiasse

Tehismõistus

Boston Dynamics: Atlas 2016
https://www.youtube.com/watch?v=rVlhMGQgDkY



DARPA robotics challenge 2015
https://www.youtube.com/watch?v=g0TaYhjpOfo

Tugev ehk lai AI:

"if a machine approaches or supersedes human intelligence, if it can do typically human tasks, if it can apply a wide range of background knowledge and has some degree of self-consciousness"

Nõrk ehk kitsas AI:

“the use of software to study or accomplish specific problem solving or reasoning tasks that do not encompass (or in some cases, are completely outside of) the full range of human cognitive abilities. “

Inimene ei teadvusta oma ajutegevust.

Inimesele paistab ekslikult, et paljud tema igapäevased toimingud on väga lihtsad ja ei nõua palju teadmisi ja ülikeerukaid ajuprotsesse

„Lihtne tehisintellekt“ ei ole võimalik.

Süsteemi võimsamaks tegemiseks on paratamatult vaja lisada üha rohkem ja rohkem erinevate spetsiaalsustega komponente.

Inimene ei ole hästi kohandunud variantide läbikaalumiseks

Arvuti on

Inimene on kohandunud uute ja ebakindlate reeglite õppimiseks

Arvuti ei ole

Filosoofia
Sotsiaalia
Tehnoloogia

Tehisintellekt meie ümber

Tehisintellekt on (vist) juba olemas: ühiskond kui suur loom paistab olema intelligentne.

Mis sellise tehisintellekti jõudu suurendab:

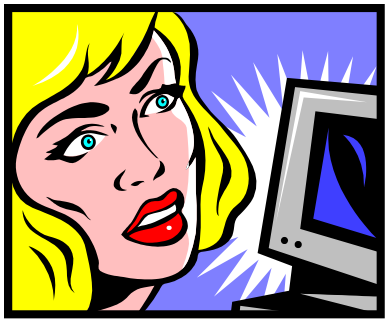
- ajurakkude (inimeste) nutikas organiseeritus
- ajurakkude (inimeste) hea kommunikatsioonivõime

Üles alla lained

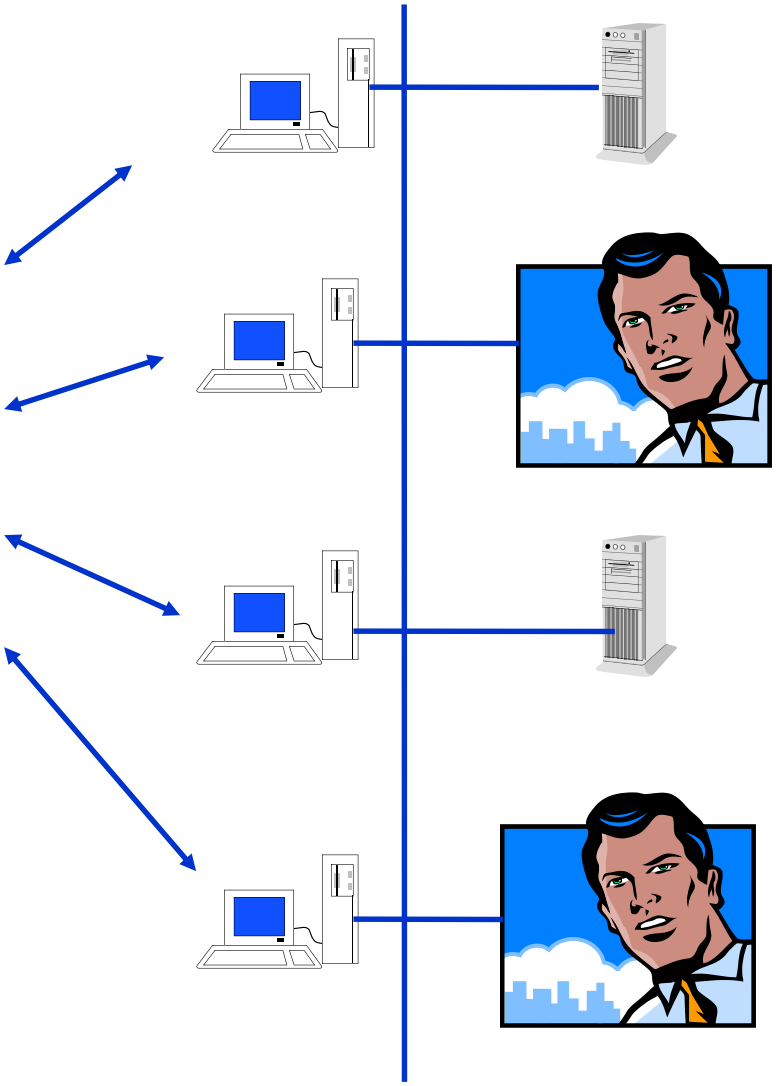
- 1952–1956: **algus**
  Mängud, lihtsad järeldussüsteemid, neurovõrgud
- 1956-1974: **kuldaeg**
  Järeldussüsteemid, loomulik keel, simulatsioonid
- 1974-1980: **esimene talv**
  Arvutusjõud väike, reaalne maailm keeruline
- 1980-1987: **uus kasv**
  Ekspertsüsteemid, teadmiste haldamine
- 1987-1995: **teine talv**
  Rakendused ainult väga väikestes nishides
- 1995 - **... Uus tõus ca 2010**

Filosoofia: üldküsimused ja Turingi test

- Turingi test: Turingi filosoofiline idee/ettepanek 30-datest:



Mõistata, kas chati-ekraani taga on inimene või programm?



Turing:
Kui katsetajad ei suuda ära arvata (st ära-arvamise sagedus on 50% ja 50% eksitakse), siis on jutlev masin **päriselt** intelligentne.

Filosoofia ja praktika Turingi testi ümber

- Palju erinevaid vastuväiteid, et testi ärategemine ei tähenda veel “päris” intellekti.
- Ükski vastuväide ei ole ümberlükatav.
- Tegelikkuses ei suuda praegu keegi teha programmi, mis suudaks Turingi testi edukalt läbi teha: sellest ollakse väga kaugel.
- Praegu “edukalt” intellekti simuleerivad Turingi testi programmid on simulaatorid, mis hoiavad keskustelu väga kitsastes raamides ja sisuliselt ei saa üldse aru talle öeldud lausetest.
- Veidi põhjalikuma vestlusega saab alati aru, et simuleerival programmil pole üldist arusaama isegi igapäevasest elust ja asjadest ning ta ei oska probleeme lahendada.

Vestlev pseudointellekt: Eliza

- In 1966 **Joseph Weizenbaum** at M.I.T. wrote **ELIZA**, a famous program that simulates a Rogerian psychoanalyst by taking excerpts from the subject's comments and posing questions back to the subject.

- Eliza on lihtne programm, mille sees on suur hulk “standardlauseid” ja “standardsõnu”. Kui Elizale anda lause, milles on sõna tema “standardsõnade” loetelus, siis Eliza valib mõne “standardvastuse” ja paneb sinna sisse osi talle antud tekstist.

- Jääb mulje, et Eliza haakub vestlejaga. Eliza poolt antud laused on samuti harilikud inimkeelsed laused (sest need on programmeerija talle andmebaasi pannud”.

Eliza: seletustega näitevestlus

ELIZA: HELLO, I'M ELIZABETH. WHAT WOULD YOU LIKE TO TALK ABOUT?

(the Welcome message)

USER: My mum is having a hard time.

ELIZA: TELL ME MORE ABOUT YOUR FAMILY

(line 5 changes 'mum' into 'mother', and this is matched at line 26; hence one of lines 28, 29, and 30 is randomly selected)

USER: There's just her, dad and my brother

ELIZA: WHAT DO YOU REMEMBER MOST ABOUT YOUR CHILDHOOD?

(line 6 changes 'dad' into 'father', and this is matched at line 27; hence either line 29 or 30 is randomly selected for display – line 28 is ruled out because it was the last response to be selected from this group)

USER: My father and brother were always quarrelling.

ELIZA: ARE YOU THE YOUNGEST IN YOUR FAMILY?

('father' is matched at line 27; hence either line 28 or 29 is randomly selected for display – line 30 is ruled out because it was the last response to be selected from this group)

USER: My brother is younger than me.

ELIZA: SO YOU ARE OLDER THAN YOUR BROTHER.

(line 31 matches, so line 32 then generates the response 'SO me IS OLDER THAN my brother'. Output substitutions then transform this in turn to 'SO YOU IS OLDER THAN my brother' (line 10), 'SO YOU IS OLDER THAN YOUR brother' (line 13), and finally 'SO YOU ARE OLDER THAN YOUR brother' (line 18).

Filosoofia ja praktika Turingi testi ümber

- Tegelikult Turingi testi edukalt tegev programm peaks:
  - Teadma igapäevaelust sama palju, kui keskmine inimene
  - Oskama lahendada ülesandeid
  - Saama aru vestluspartneri soovidest ja teadmistest jne
- See kõik on väga keeruline ja ei ole põhjust arvata, et lihtsakoeline programm sellega kuidagi hakkama saab.
- Kui programm sellega hakkama saab, siis on ta äärmiselt keeruline ja inimesed üldiselt ei saa enam aru, mis ta sees on ja kuidas ta töötab, ja pigem võiks arvata, et ta on siis ka päriselt intelligentne.

Filosoofia ja praktika Turingi testi ümber

Näited arusaamise raskustest:

The town councillors refused to give the angry demonstrators a permit because they feared violence. Who feared violence?

a) The town councillors
b) The angry demonstrators

The large ball crashed right through the table because it was made of Styrofoam. What was made of Styrofoam? (The alternative formulation replaces Stryrofoam with steel.)

a) The large ball
b) The table

Äärmused: putukad ja CYC

- **Putukad**

  - Rodney Brooks'i projekt MIT-s:
    teha putuka analoogideks olevaid roboteid.

  - Idee: kui oskame teha putuka intellektiga masinat, siis proovime sealt edasi liikuda väikese looma intellektiga masinani, sealt kõrgema loomani, sealt inimeseni ja edasi.
  - Lootus: sel viisil ehitame mõistuse mehhanismid analoogiliselt nende tekkele looduses.

- **CYC**

  - Doug Lenati projekt CYC: teeme hiiglasliku andmebaasi faktidest ja reeglitest ja lihtsa järelduste tegemise programmi sinna peale.
  - Idee: programm hakkab tekstidest aru saama, uusi reegleid ja fakte õppima, seejärel ka iseennast täiustama.
  - Lootus: kvantiteet tekitab teatud piiri ületamisel kvalitatiivse hüppe.

Suured rakendusvaldkonnad

Pildituvastus
Inimkeelest arusaamine
Ülesannete lahendamine
Robootika

Meetodid

Otsing
Õppimine

Otsing

Keegi peab lubatud käigud või reeglid kirja panema.

Aga kuidas? Millised reeglid?

Otsingu koolkonnad

Fikseeritud reeglid

Formaliseeritud reeglisüsteemid

Fikseeritud reeglid

Kõik väga OK, kui seisust vähe variante a la kabe. Kukub AI valdkonnast välja.

Kõik halvasti, kui seisust väga palju variante a la Go või matemaatika.

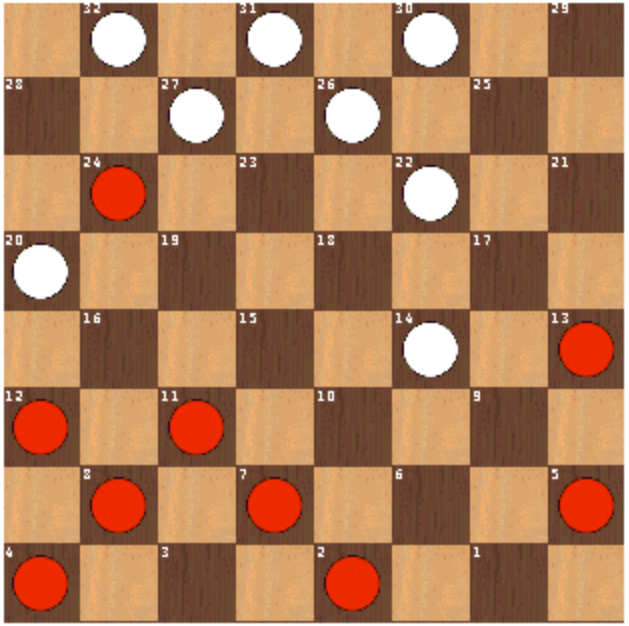
1997

IBM Deep Blue võidab Kasparovi

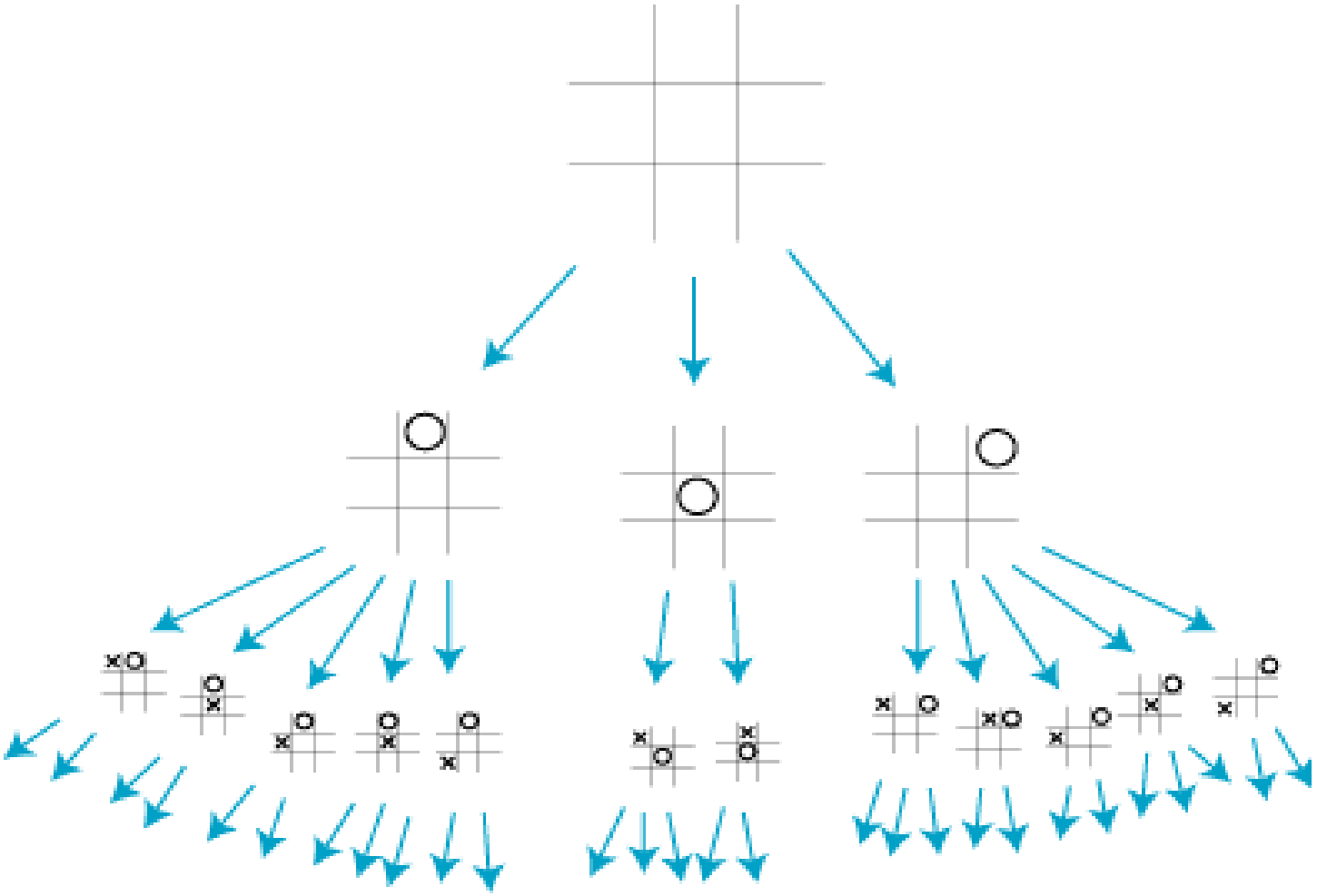
Inglise kabe lahendatud: 2007, Chinook UAlberta

Checkers has a search space of $5x10^{20}$

Almost continuously since 1989, dozens of computers had been working around the clock to solve the game.



Mänguvariantide puu



Kuidas maleprogrammi teha?

- Teeme kõigepealt programmi, mis:
  - loeb seisu
  - teeb seisu järgi mällu tabeli kõigist võimalikest käikudest selles seisus.
- See ei olegi nii väga keeruline programm
  - Males keerulisem
  - Kabes lihtsam
  - Othello (reversi) veel lihtsam
  - Viis nuppu ritta: väga lihtne
- Nüüd võib programm tabelist juhusliku käigu valida.
- Ja juba mängibki! Kuigi kehvasti ....

Kuidas programmi veidi paremaks teha?

- Teeme eraldi väikese programmi, mis:
  - Võtab seisu ette
  - Ütleb umbes, kui hea seis on.
  - Seisu “headus” on number, mille meie programm arvutab.
- Kuidas öelda “kui hea”?
  - Vahel lihtne: kaotus / võit/ ei kumbki.
  - Numbrid: 1 -1 0
  - Keerulisem: “harilik maleseis”. Kumbalgi pole võitu ega kaotust.

Seisu hindamine

- Keerulisem olukord: harilik maleseis.
- Materjal:
  - Loeme kokku meie ja vastase nupud.
  - Lipp annab 10 punkti, vanker 6, jne.
- Positsioon:
  - Loeme kokku eelviimasel real etturid masinal/inimesel
  - Loeme kokku tsentriväljadel olevad etturid masinal/inimesel
  - Loeme kokku, mitmele tsentriväljale masin/inimene tuld annab
  - Vaatame üle kuninga kaitse tugevuse
  - .... jne .....

- Kokku saame seisu headuse numbri:
  - Masina materjal – inimese materjal + masina positsioon – inimese positsioon

Mis edasi?

- Nüüd on meil kõigi käikude tabel, iga käigu juures arvutame tekkiva seisu headuse numbri.
- Valime kõige parema tulemuse (maksimumi numbritest!) andva käigu.
- Programm hakkab veidi paremini mängima, aga:
  - Ikka mängib väga kehvasti!
  - Miks? Sest meie seisu headuse hindamise programm on nigel.
  - Kuidas headuse hindamist parandada?

- Loomulikult tuleks vaadata, mis käike vastane võib peale meie käike teha! Ja kuidas meie võime vastata. Jne.

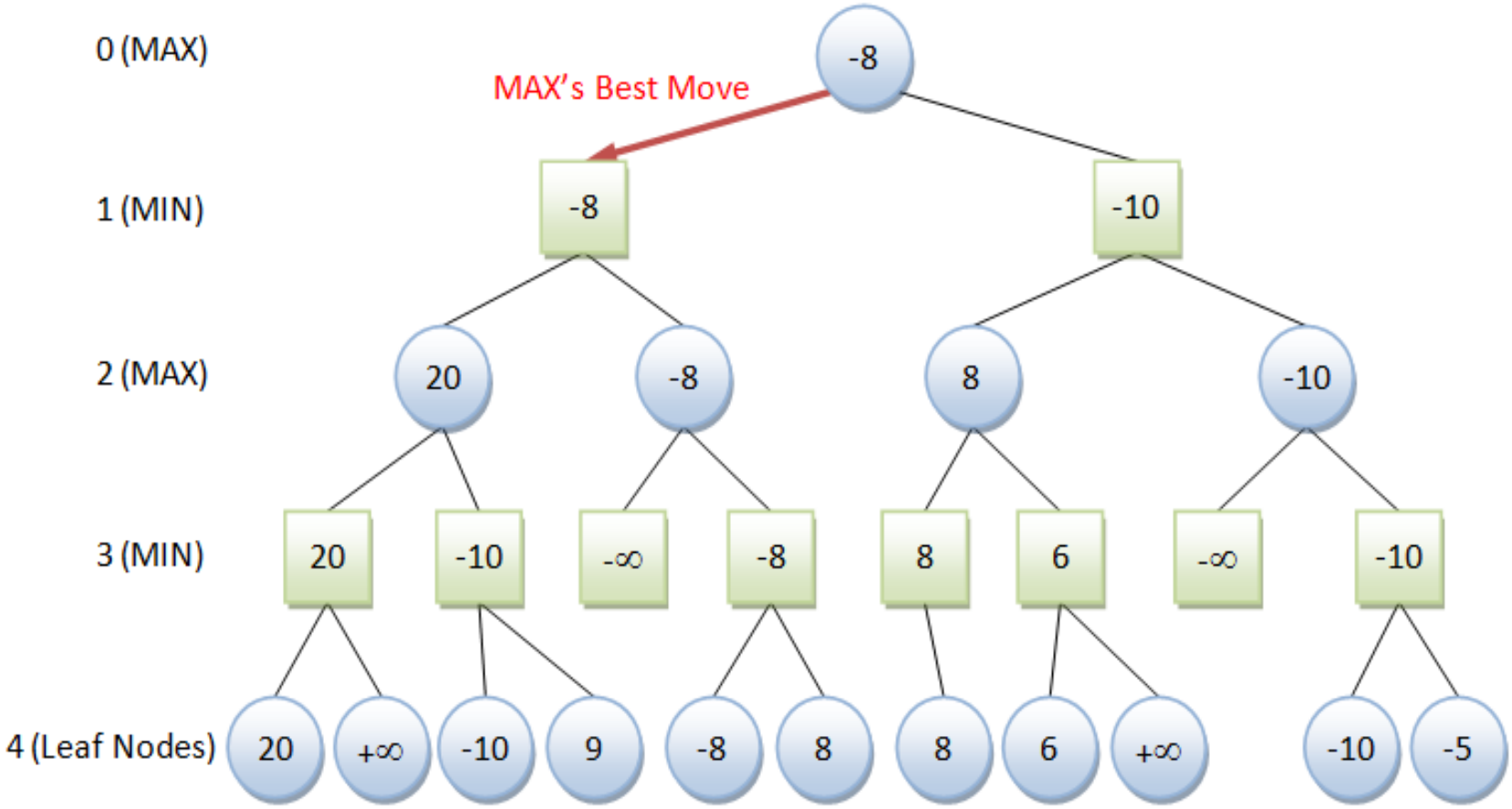
Kuidas sellist seisude puud kasutada?

- Eeldame lihtsalt, et:
  - masin tahab teha käiku, mis annab kõige suurema headuse numbriga seisu.
  - vastane tahab teha käiku, mis annab kõige väiksema headuse numbriga seisu.

- Seega:
  - igast seisust valib masin käigu, mis on maksimum-headusega (masinale).
  - igast seisust valib vastane käigu, mis on miinimum-headusega (masinale).

- Idee:
  - Vaatame käikude puud sügavuseni N (näiteks N=3)
  - Kõige alumistel seisudel arvutame lihtsalt headuse välja
  - Seejärel “tõstame” headuse numbreid ülespoole!

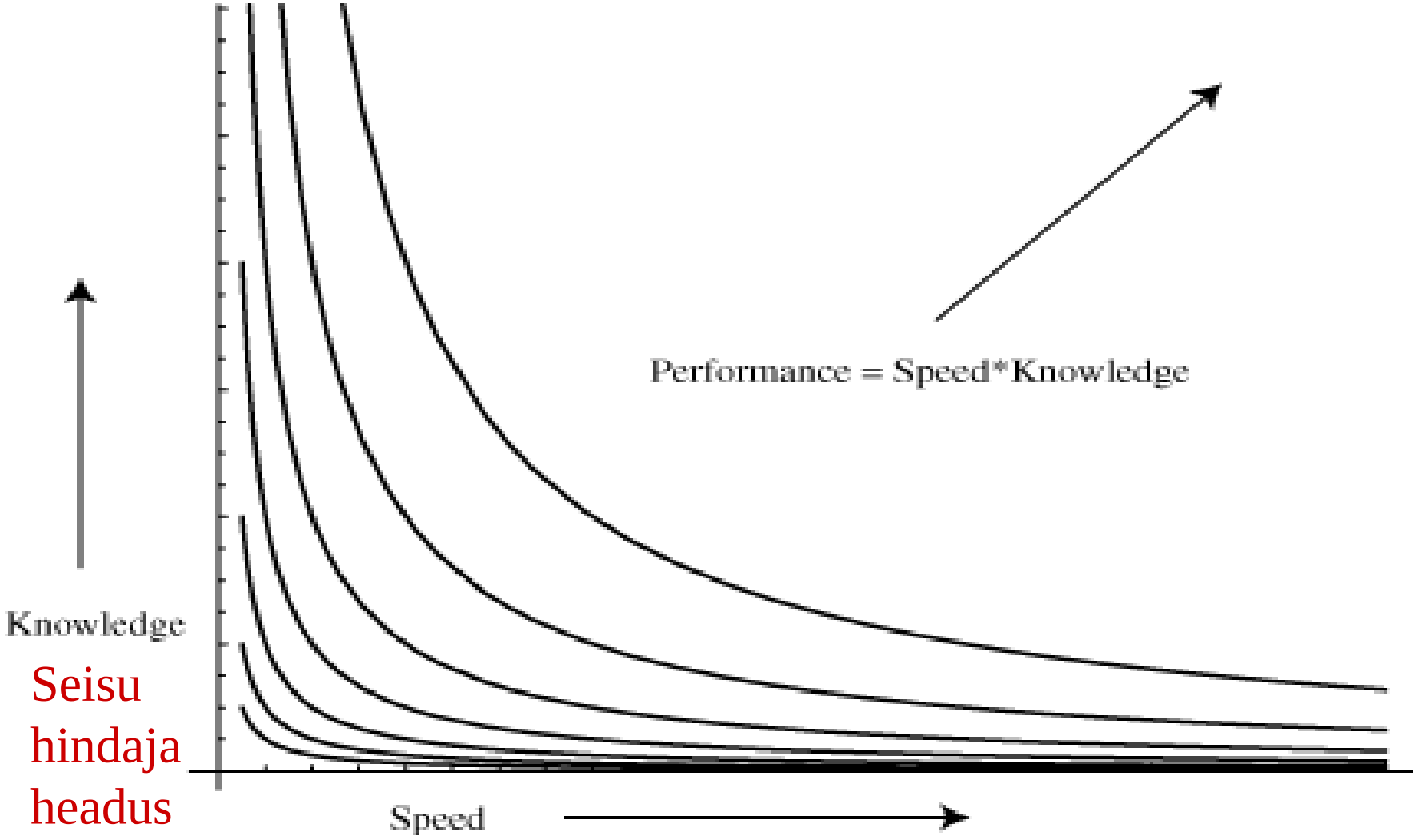
Otsing mängupuust: minimax algoritm



Kui heaks programmi saab?

- Mida sügavamat puud masin läbi jõuab vaadata, seda täpsemini ta käiku oskab valida.
- Puu läheb kiiresti väga suureks!
- Males ca 30 käiku ühes seisus.
  - **Esimesel tasemel käike 30.**
  - **Teisel tasemel käike 30*30**
  - **Kolmandal tasemel käike 30*30*30**
  - **....**
  - **N-ndal tasemel käike 30 astmes N.**
- Viiekümnendal tasemel oleks käike ca 30 astmes 50. See on rohkem, kui elementaarosakesi universumis!

Umbes nii: teadmised ja kiirus ja mängu headus



Seisu hindaja headus

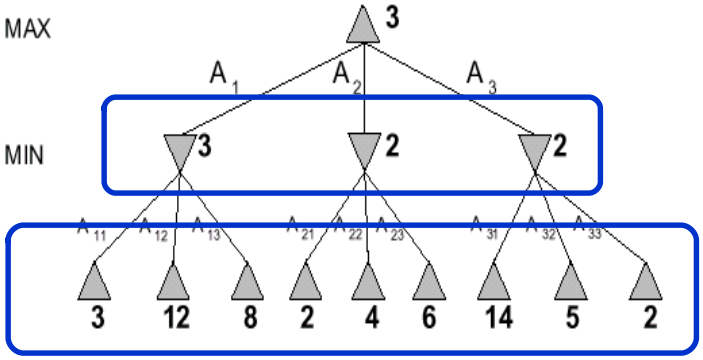
Kui suure puu jõuab läbi vaadata

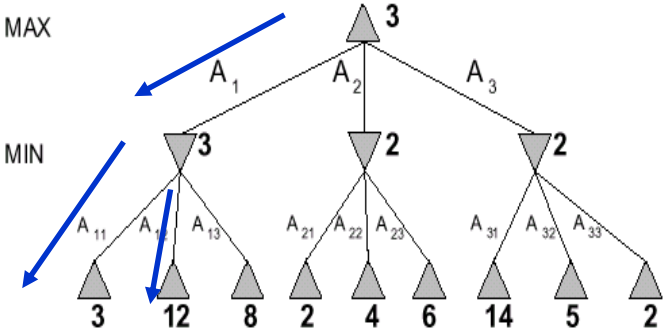
Kuidas programmi parandada?

- Ei ole ühte head lahendust. On palju erinevaid nõkse!
- Näiteks:
  - Teeme seisu hindaja paremaks (programmi “targemaks”).
  - Aga siis läheb ta aeglasemaks ka.
  - Seega jõuame vähem käike läbi vaadata.
- Enamik nõkse on seotud käikude puu vähendamisega: ei ole vaja kogu puud läbi vaadata.

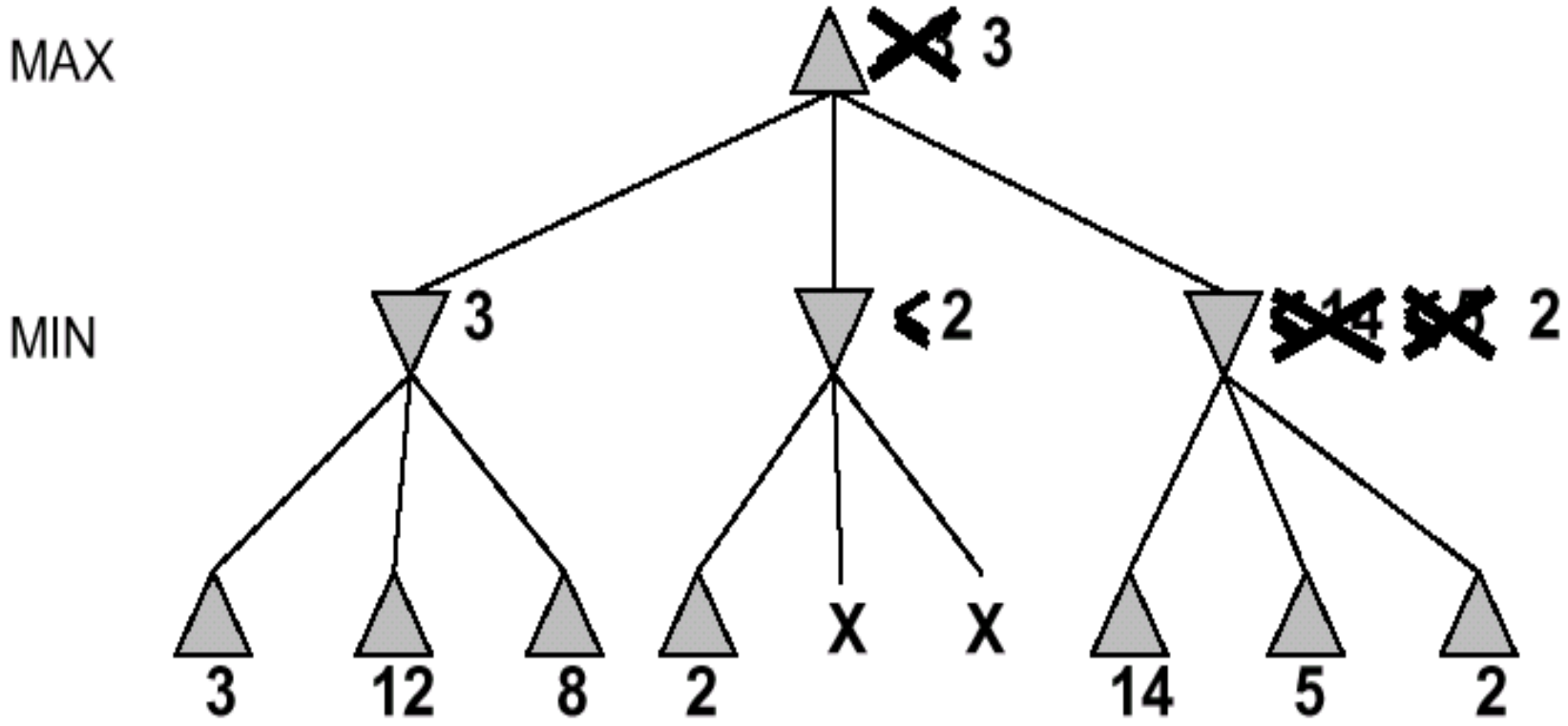
Laiuti vs sügavuti otsing

- Kaks võimalust:
  - Otsime puu läbi kiht-kihilt
  - Otsime puu läbi sügavuti, minnes algul vasakul maksimaalse sügavuseni
- Eelistatakse sügavuti otsingut!
  - Mälu vaja palju vähem
  - Muud eelised ka





Üks esimesi universaalseid meetodeid: alpha-beta



Koodinäide

```
int AlphaBeta (pos, depth, alpha, beta)
{
    if (depth == 0) return Evaluate(pos);
    best = -INFINITY;
    succ = Successors(pos);
    while (not Empty(succ) && best < beta)
    {
        pos = RemoveOne(succ);
        if (best > alpha) alpha = best;
        value = -AlphaBeta(pos, depth-1, -beta, -alpha);
        if (value > best) best = value;
    }
    return best;
}
```

Soteerimise nõks otsingu kiirendamiseks

- Sorteerida variandid seisust X enne otsingut ära: alustada tõenäoliselt paremate käikude proovimisega.
  - Suurendab tohutult alpha-beta efekti!
  - Kuidas sorteerida?
- Iteratiivne süvenemine. Teeme:
  - algul täisotsingu sügavuseni 2,
  - siis uue täisotsingu sügavuseni 4,
  - siis uue täisotsingu sügavuseni 6,
  - .... jne ....
  - iga kord kasutame eelmise otsingu tulemust sorteerimiseks!

Muud standardnõksud

- "**Killer moves**": jätame otsides meelde eriti head käigud:
  - nii masinal kui vastasel
  - proovime kõigepealt varasemast meelde jäetud eriti häid käike
- "**Quiescence search**": mõnda haru otsitakse sügavamalt:
  - ebastabiilses seisus otsime sügavamalt
  - stabiilses seisus otsime vähem sügavalt
  - otsime lõpuni kõik vahetused ja löögid
- **"Null-Move"**: mis siis, kui vastane saaks kaks käiku järjest?
  - Proovime nii, et vastane saab kaks käiku järjest
  - Kui on meile OK tulemus, siis see on positiivne faktor
  - Kui on meile halb tulemus, jätame meelde "killer move"

Kuidas mõjub?

- **Mida sügavam puu, seda suurem mõju. Näiteks:**
- **Käikude puu sügavus viis:**
  - MiniMax: hindab 10,541,242 seisu
  - Alpha-Beta: hindab 1,037,209 seisu
  - A-B + “killer moves”: 530,587 seisu.
- **Käikude puu sügavus seitse:**
  - MiniMax: hindab ca 8,100,000,000 seisu
  - Alpha-Beta: hindab 162,662,568 seisu
  - A-B + “killer moves”: 46,455,262 seisu.

Täiendav idee: lõppmängude andmebaasid

- **Idee:**
  - ehitame hiiglasliku lõppmängu-seisude andmebaasi
  - igal lõppmängu-seisul on andmebaasis öeldud täpne “headus” (võit, viik, kaotus)
  - ehitamiseks: kõigepealt ühe nupuga lõppmängud, siis kahe nupuga lõppmängud, siis kolme nupuga lõppmängud jne.

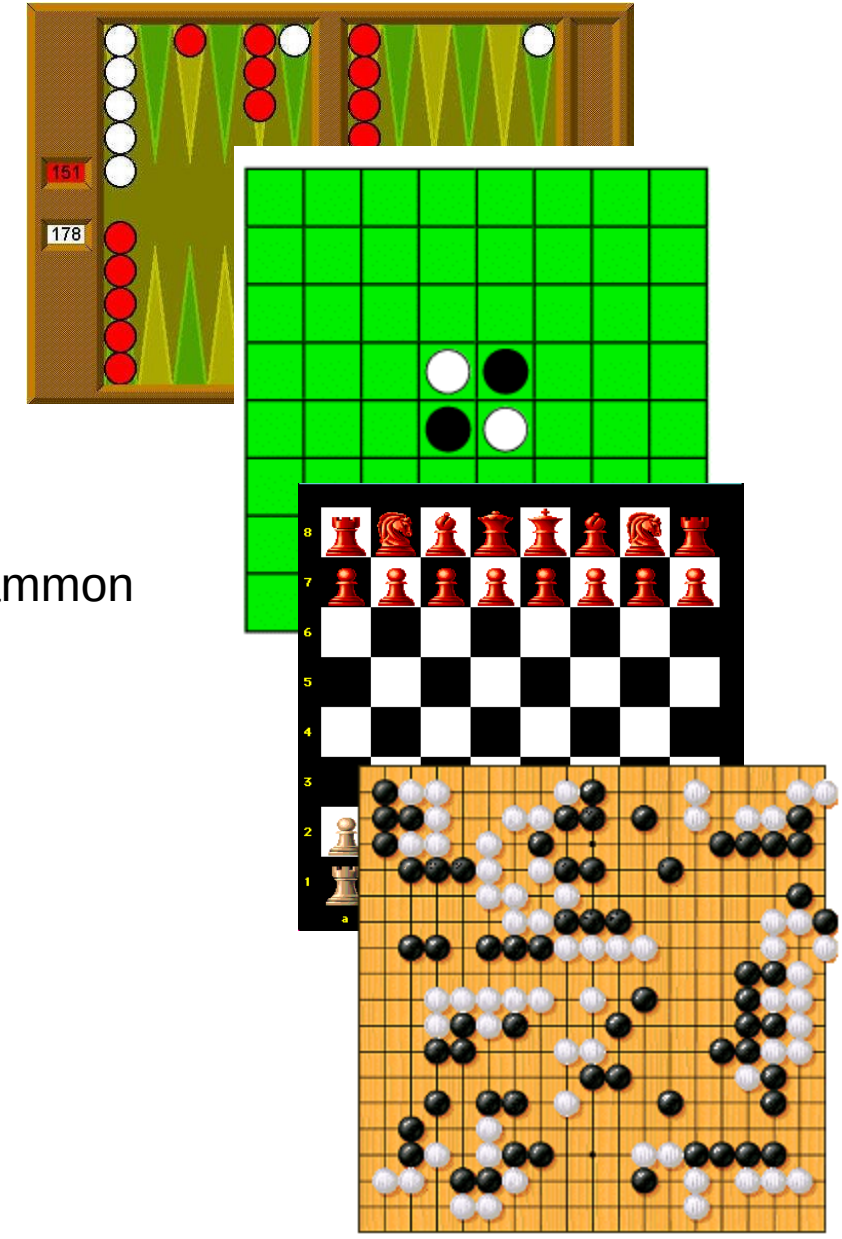
- **Kõigepealt tehti seda 8x8 inglise kabes:**
  - **Chinook**: Jonathan Schaeffer, Robert Blake, Paul Lu and Martin Bryant: University of Alberta
  - Kõigi seisude andmebaas, kus on 10 või vähem nuppu

- **Sügavad otsingud jõuavad tihti lõppmängude andmebaasini, kus on juba sees täpne seisuhinnang.**

- **Otsing seega mõlemast suunast!**

Praegune seis mõtlemismängudega

- Lahendatud:
  - Neli nuppu ritta, Qubic,
  - Nine Man's Morris, Go-Moku,
  - Awari
  - Inglise kabe (8x8)
- Väga tugevalt üle inimmängijate:
  - Renju (viis nuppu ritta),
  - Othello (Reversi), Scrabble, Backgammon
- Veidi üle maailmameistri taseme
  - Male,
  - Rahvusvaheline kabe (10x10)
- Maailmameistri tasemel :
  - Go (19x19)
- Tippmängija tasemel (???):
  - Bridzh, Pokker



Formaliseeritud reeglisüsteemid

Kõik väga OK, kui õnnestub kirja panna väike hulk selgeid reegleid.

Kõik halvasti, kui reegleid on palju:

- Ei jõua/oska neid kirja panna.
- Ei jõua otsida.

Lahendamine loogikatõestuse abil

- Ülesanne tuleb loogika keeles formuleerida.
- Küsimus tuleb ka loogika keeles formuleerida.
- Tõestaja asub tõestust otsima.

- Kui anda lõpmatult palju aega ja mälu, siis tõestaja lõpuks ka tõestuse leiab (kui tõestus üldse teoreetiliselt eksisteerib). Keeruliste ülesannete puhul võtaks see lootusetult kaua aega (meenuta keerukusklasse!)

- Kui tõestust ei ole, siis tõestaja enamasti jääbki seda otsima, teadma, et sellist tõestust ei saa olla.

Masinaga matemaatikaprobleem

W. McCune 1996:

- The Robbins problem---are all Robbins algebras Boolean?- has been solved using his automated theorem prover EQP.
- Programm otsis lahendust ca üks nädal, kuni lõpuks leidis.
- Ülesanne oli matemaatikute poolt lahendamata, kuigi püstitati aastal 1933:

- Meil on antud Robbinsi algebra:

x + y = y + x.	[commutativity]
(x + y) + z = x + (y + z).	[associativity]
n(n(x + y) + n(x + n(y))) = x.	[Robbins equation]

- Kas järgmised võrdused annava selle algebra jaoks vastuolu:

x+y != x.
n(x+y) != n(x).

Tõestuse algus

2 [] x+y=y+x.
3 [] (x+y)+z=x+y+z.
4 [] (x+y)+z=x+y+z.
5 [] n(n(n(x)+y)+n(x+y))=y.
6 [] n(x+y)!=n(x).
64 [para_into,4.1.1.1,2.1.1,demod,3] x+y+z=y+x+z.
71 [para_into,5.1.1.1.2.1,2.1.1] n(n(n(x)+y)+n(y+x))=y.
73 [para_into,5.1.1.1,2.1.1] n(n(x+y)+n(n(x)+y))=y.
75 [para_into,6.1.1.1,2.1.1] n(x+y)!=n(y).
93 [para_into,71.1.1.1,2.1.1] n(n(x+y)+n(n(y)+x))=x.
116 [para_into,75.1.1.1,4.1.1] n(x+y+z)!=n(z).
130 [para_into,93.1.1.1.2,73.1.1] n(n(n(n(x)+y)+x+y)+y)=n(n(x)+y).
132 [para_into,93.1.1.1.2,5.1.1] n(n(n(x+y)+n(x)+y)+y)=n(x+y).
139 [para_into,116.1.1.1.2,64.1.1] n(x+y+z+u)!=n(y+u).
170 [para_from,130.1.1,73.1.1.1.2,demod,3,3] n(n(n(n(x)+y)+x+y+y)+n(n(x)+y))=y.
174 [para_from,130.1.1,5.1.1.1.1,demod,3,3] n(n(n(x)+y)+n(n(n(x)+y)+x+y+y))=y.
190 [para_into,132.1.1.1.1.1.2,64.1.1] n(n(n(x+y+z)+y+n(x)+z)+y+z)=n(x+y+z).
211 [para_into,139.1.1.1,64.1.1] n(x+y+z+u)!=n(x+u).
253 [para_from,170.1.1,93.1.1.1.2] n(n(n(n(x)+y)+n(n(x)+y)+x+y+y)+y)=n(n(x)+y).
260 [para_from,170.1.1,73.1.1.1.2.1.1,demod,3] n(n(n(n(n(x)+y)+x+y+y)+n(n(x)+y)+z)+n(y+z))=z.
346 [para_into,190.1.1.1.2,2.1.1] n(n(n(x+y+z)+y+n(x)+z)+z+y)=n(x+y+z).
....

Tõestus jätkub

....
423 [para_from,253.1.1,5.1.1.1.1,demod,3,3,3,3] n(n(n(x)+y)+n(n(n(x)+y)+n(n(x)+y)+x+y+y+y))=y.
475 [para_from,260.1.1,73.1.1.1.2,demod,3,3] n(n(n(n(n(x)+y)+x+y+y)+n(n(x)+y)+z+n(y+z))+z)=n(y+z).
486 [para_into,346.1.1.1.1.1.1.1,2.1.1,demod,3] n(n(n(x+y+z)+x+n(z)+y)+y+x)=n(z+x+y).
684 [para_into,475.1.1.1.1.1.2.2,2.1.1] n(n(n(n(n(x)+y)+x+y+y)+n(n(x)+y)+n(y+z)+z)+z)=n(y+z).
801 [para_from,684.1.1,73.1.1.1.2.1.1,demod,3]
n(n(n(n(n(n(x)+y)+x+y+y)+n(n(x)+y)+n(y+z)+z)+z+u)+n(n(y+z)+u))=u.
848 [para_into,801.1.1.1.1,486.1.1,demod,3] n(n(x+x+n(n(x)+x)+x)+n(n(x+x)+n(n(x)+x)))=n(n(x)+x).
878 [para_into,848.1.1.1.2,93.1.1] n(n(x+x+n(n(x)+x)+x)+x)=n(n(x)+x).
897 [para_into,878.1.1.1.1.1.2,64.1.1] n(n(x+n(n(x)+x)+x+x)+x)=n(n(x)+x).
948 [para_into,897.1.1.1.1.1,64.1.1] n(n(n(n(x)+x)+x+x+x)+x)=n(n(x)+x).
1015 [para_from,948.1.1,73.1.1.1.2.1.1,demod,3] n(n(n(n(n(x)+x)+x+x+x)+x+y)+n(n(n(x)+x)+y))=y.
1036 [para_from,948.1.1,73.1.1.1.2,demod,3,3,3] n(n(n(n(x)+x)+x+x+x+x+x)+n(n(x)+x))=x.
1053 [para_into,1015.1.1.1.2,423.1.1]
n(n(n(n(n(x)+x)+x+x+x)+x+n(n(n(x)+x)+n(n(x)+x)+x+x+x+x))+x)=n(n(n(x)+x)+n(n(x)+x)+x+x+x+x).
1079 [para_from,1036.1.1,93.1.1.1.2] n(n(n(n(x)+x)+n(n(x)+x)+x+x+x+x)+x)=n(n(x)+x).
1112 [para_into,1053.1.1.1.1.1,2.1.1,demod,3]
n(n(x+n(n(n(x)+x)+n(n(x)+x)+x+x+x+x)+n(n(n(x)+x)+x+x+x+x))+x)=n(n(n(x)+x)+n(n(x)+x)+x+x+x+x).
1130 [para_from,1079.1.1,73.1.1.1.2.1.1,demod,3] n(n(n(n(n(x)+x)+n(n(x)+x)+x+x+x+x)+x+y)+n(n(n(x)+x)+y))=y.
1149 [para_into,1112.1.1.1.1.1,64.1.1]
n(n(n(n(n(x)+x)+n(n(x)+x)+x+x+x+x)+x+n(n(n(x)+x)+x+x+x+x))+x)=n(n(n(x)+x)+n(n(x)+x)+x+x+x+x).
1169 [para_into,1130.1.1.1.2,174.1.1]
n(n(n(n(n(x)+x)+n(n(x)+x)+x+x+x+x)+x+n(n(n(x)+x)+x+x+x+x))+x)=n(n(n(x)+x)+x+x+x).
1211 [para_into,1169.1.1,1149.1.1] n(n(n(x)+x)+n(n(x)+x)+x+x+x+x)=n(n(n(x)+x)+x+x+x).
1212 [binary,1211.1,211.1] $F.

Ekspertsüsteemid

- Ekspertsüsteem on mõne spetsiifilise valdkonna jaoks kohandatud järelduste tegemise programm (variant teoreemitõestajast)
- Tihtpeale kasutavad ebaharilikku, spetsiaalselt valdkonnaga sobitatud loogikat.
- Enamasti sisaldavad paljusid valmiskujul reegleid.
- Enamasti sisaldavad mugavat kasutajaliidest: kasutaja ei pea loogika keelt oskama.
- Tüüpiliselt kasutatakse mõne suure rakenduse osana

Loogika ja ebakindlad teadmised

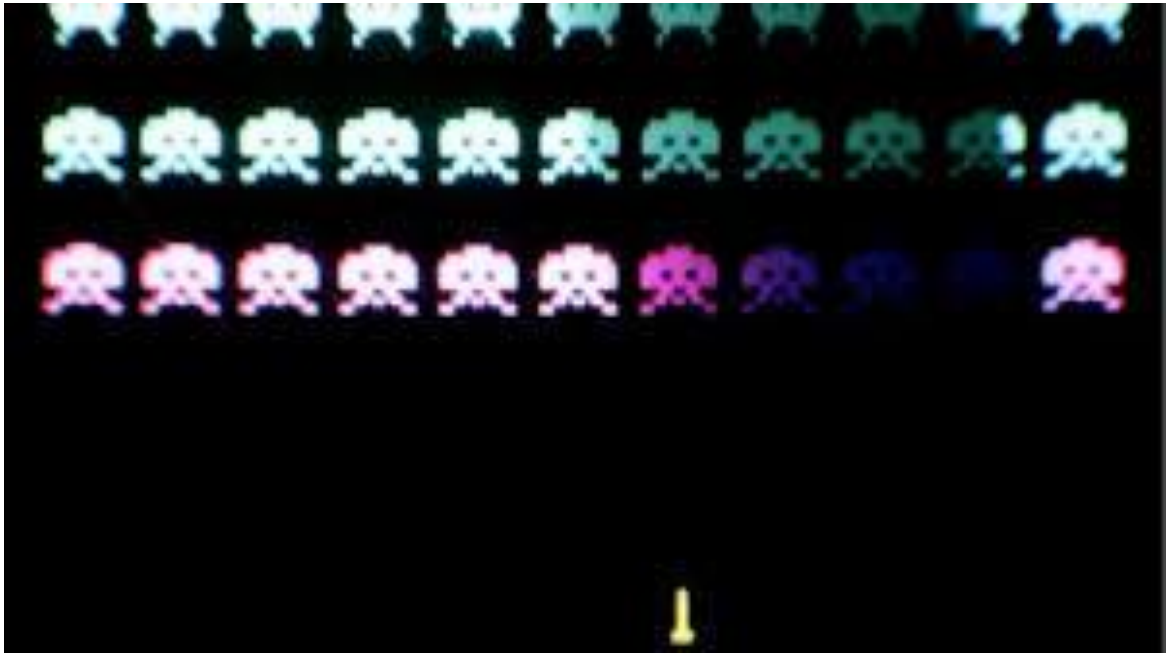
Suur hulk väga erinevaid tõenäosuslikke loogikaid.

Väga vähe reaalselt töötavaid süsteeme.

Õppimine

Loome reeglid ettantud näidetest.

Kust saada palju näiteid? Millised reeglid? Mida nende reeglitega teha saab?



2015

Google Deep Mind õpib mängima Atari videomänge

2016

Google Deep Mind võidab Go tshempionit

Õppimise koolkonnad

Statistika

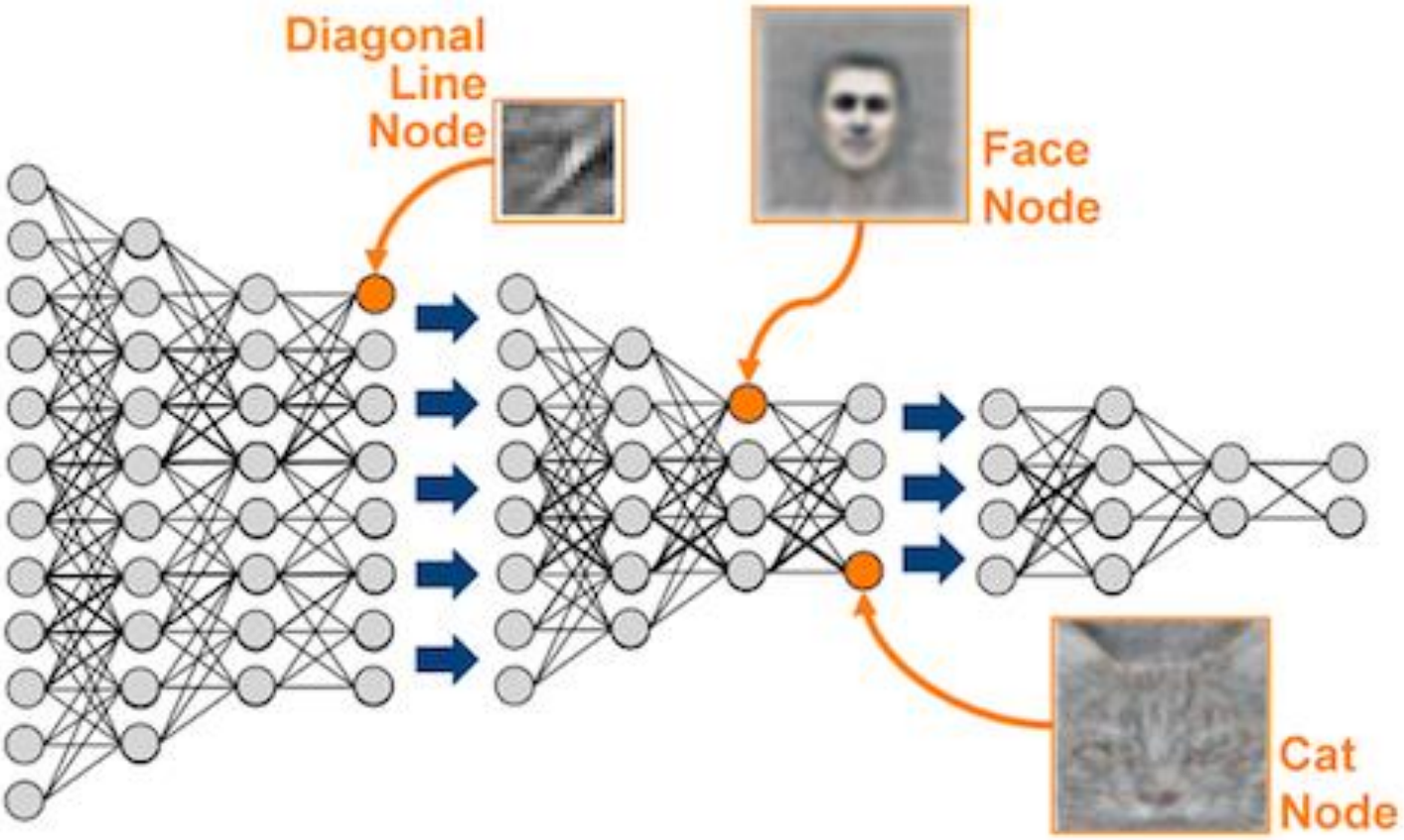
Neurovõrgud

Statistika

Kõik OK, kui arendaja saab ise umbes aru, mis toimub. Statistika optimeerib inimese mõeldud valemis parameetreid.

Kõik halvasti, kui ei oska valemeid välja mõelda.

Neurovõrgud



Neurovõrgud

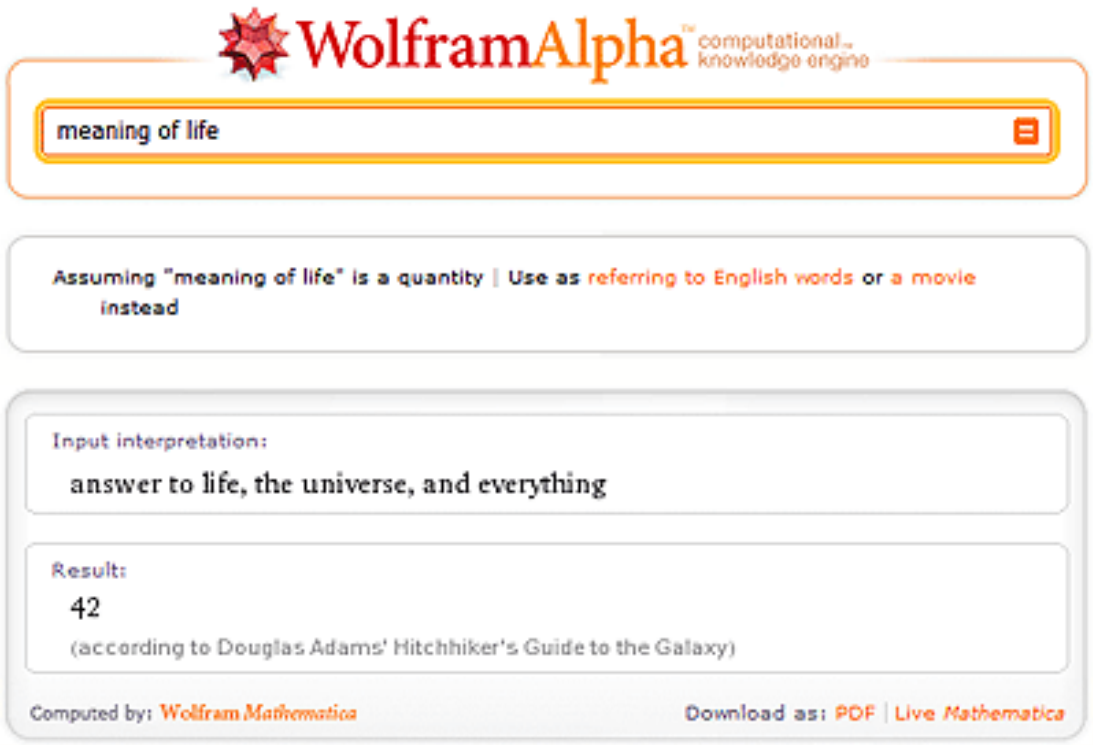
Kõik OK, kui variantide arv on abstraheeritav üpris piiratuks.

Kõik halvasti, kui variante liiga palju, näiteid liiga vähe, sisulised reeglid osutuvad keeruliseks.

Koolkondade vahealad

Ebakindlad reeglid
Reeglite õppimine
Õpitud meetodite seletamine

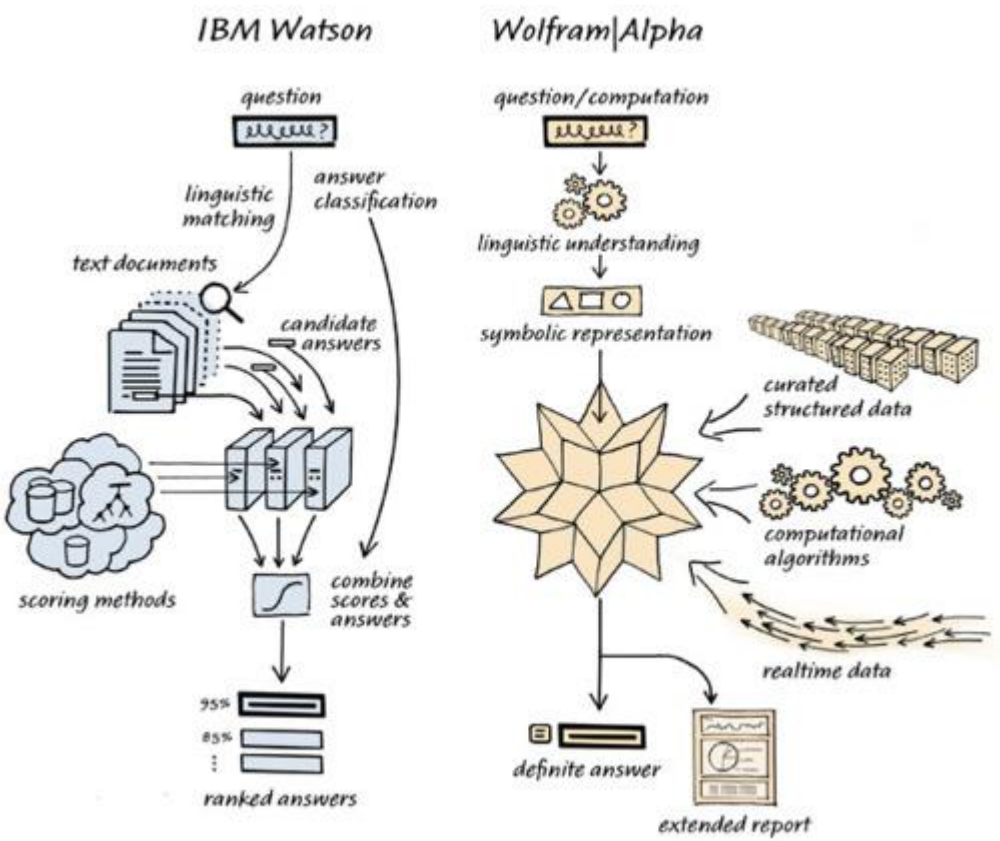
Wolfram alpha online: 2009



2011

IBM Watson võidab Jeopardy mälumängu

Watson ja Wolfram alpha



Robootika

- Sisuliselt süntees kogu tehisintellektindusest.
- Lisandub:
  - Mehaanika
  - Mehaanika kiire ja täpne juhtimine (füüsikalised arvutused)
- Seepärast on “vingeid” roboteid praegusaja tehnoloogiaga pea võimatu teha.
- Näiteks ei suuda keegi teha robot-tennisisti, kes natukenegi mängiks.
- Tehtud on (suhteliselt kehvasid) robot-lauatennisemängijaid.
- TTÜ projekt: roboswarm.eu
- Tipptasemel robotid: Boston dynamics „big dog“ ja Darpa grand challenge autod.

2005

Stanfordi Stanley võidab DARPA Grand Challenge II

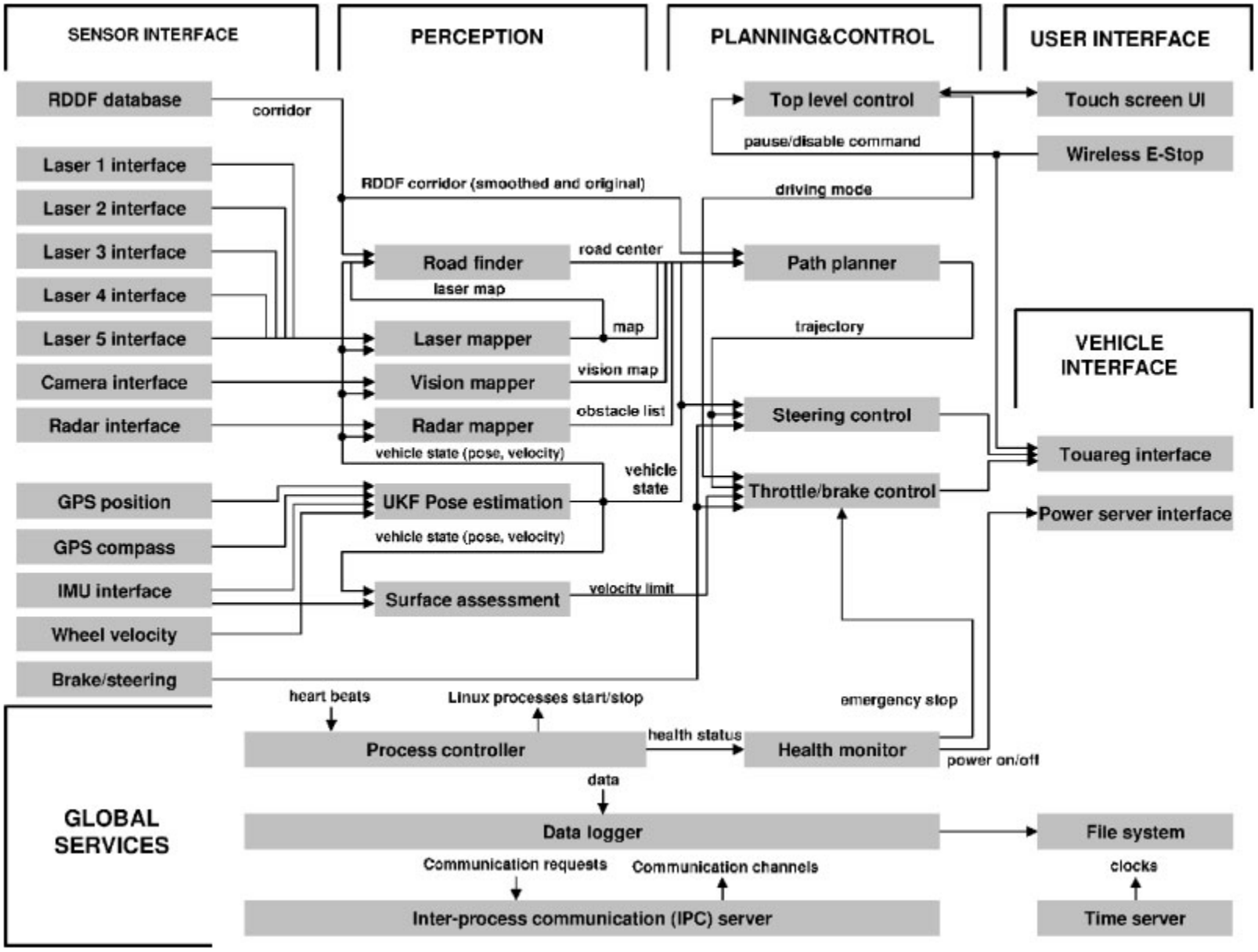
Darpa urban challenge 2007

96 km linnas, 4:10

Google autonomous car:

Based on vehicle Stanley which won the 2005 DARPA Grand Challenge.

In August 2012, the team announced that they have completed over 300,000 autonomous-driving miles (500 000 km) accident-free


SENSOR INTERFACE
PERCEPTION
PLANNING&CONTROL
USER INTERFACE
RDDF database
corridor
Laser 1 interface
Laser 2 interface
Laser 3 interface
Laser 4 interface
Laser 5 interface
Camera interface
Radar interface
GPS position
GPS compass
IMU interface
Wheel velocity
Brake/steering
RDDF corridor (smoothed and original)
Road finder
road center
laser map
Laser mapper
map
Vision mapper
vision map
Radar mapper
obstacle list
vehicle state (pose, velocity)
UKF Pose estimation
vehicle state
vehicle state (pose, velocity)
Surface assessment
velocity limit
Top level control
pause/disable command
driving mode
Path planner
trajectory
Steering control
Throttle/brake control
Touch screen UI
Wireless E-Stop
VEHICLE INTERFACE
Touareg interface
Power server interface
heart beats
Linux processes start/stop
emergency stop
Process controller
health status
Health monitor
power on/off
data
GLOBAL SERVICES
Data logger
File system
Communication requests
Communication channels
clocks
Inter-process communication (IPC) server
Time server